TECNOLOGICO NACIONAL DE MEXICO

INSTITUTO TECNOLOGICO DE ORIZABA

Asignatura: Estructura de Datos

Carrera: Ingeniería en Informática

Estudiantes:

- Hernandez Angón Diego - No. Control 21010193
- Dorantes Rodríguez Diego Yael - No. Control 21010184

Profesora: María Jacinta Martínez Castillo

Grupo: 3a3A

Fecha de entrega: 22/04/2023

Reporte de unidad 3: Estructuras Lineales

INTRODUCCION

En este 3er reporte de la unidad, abordaremos el tema de las estructuras lineales, conformadas por pilas, colas, y listas enlazadas. Hablaremos sobre la definición de pilas y colas, además de sus aplicaciones. Utilizaremos tanto memoria dinámica como memoria estática a lo largo del desarrollo de los códigos y aprenderemos acerca de cómo funcionan y gestionan los datos las pilas y colas. La pila es una estructura de datos LIFO (Last In, First Out), lo que significa que el último elemento que se agrega a la pila es el primero en salir, una cola es una estructura de datos que se utiliza para almacenar elementos en una secuencia ordenada. La cola es una estructura de datos FIFO (First In, First Out), lo que significa que el primer elemento que se agrega a la cola es el primero en salir.

COMPETENCIA ESPECIFICA

Desarrollar la habilidad para el desarrollo de código, utilizando a las pilas y colas como estructuras de datos para, en un futuro aplicarlas correctamente, sea en el ámbito laboral o personal. Identificar las características de las pilas y colas, sus diferencias, ventajas y desventajas, saber como crear una interfaz e implementarla en códigos junto con los métodos push, pop, free y peek.

MARCO TEORICO

- Clases genéricas y arreglos dinámicos

Las clases genéricas y los arreglos dinámicos son conceptos relacionados en la programación que se utilizan para trabajar con colecciones de elementos de cualquier tipo de datos. Las clases genéricas se utilizan para diseñar clases que funcionen con cualquier tipo de dato, mientras que los arreglos dinámicos se utilizan para almacenar y acceder a colecciones de elementos de forma dinámica en tiempo de ejecución. Ambas herramientas son muy útiles en la programación y se utilizan en muchos proyectos de software.

Pilas

Las pilas son estructuras de datos que se utilizan en programación para almacenar y gestionar un conjunto de elementos. Las pilas se caracterizan por ser una estructura de datos lineal, en la que los elementos se organizan en una secuencia, y por seguir el principio "last-in, first-out" (LIFO), lo que significa que el último elemento que se agrega a la pila es el primero que se elimina.

Las pilas se pueden implementar de varias formas, como mediante el uso de arreglos estáticos o dinámicos, o mediante el uso de listas enlazadas. En general, la implementación de una pila se basa en la manipulación de la parte superior de la pila (que se llama "puntero") y en la actualización del puntero después de cada operación.

Las pilas son una estructura de datos muy útil en programación que se caracterizan por seguir el principio LIFO y que se utilizan para almacenar y gestionar un conjunto de elementos. Sus aplicaciones:

- Validación de paréntesis: Las pilas se utilizan para verificar si una expresión matemática o una expresión regular tiene una sintaxis correcta en cuanto a la apertura y cierre de paréntesis.
- Inversión de cadenas: Las pilas se utilizan para invertir el orden de los caracteres en una cadena.
- Implementación de la navegación hacia atrás en los navegadores web: Las pilas se utilizan para mantener un registro de las páginas web que se han visitado para permitir la navegación hacia atrás.
- Gestión de la memoria: Las pilas se utilizan en los sistemas operativos para gestionar la memoria de un programa.

Colas

Las colas son estructuras de datos en programación que se utilizan para almacenar y gestionar un conjunto de elementos. Las colas se caracterizan por ser una estructura de datos lineal, en la que los elementos se organizan en una secuencia, y por seguir el principio "first-in, first-out" (FIFO), lo que significa que el primer elemento que se agrega a la cola es el primero que se elimina.

Una cola se puede entender como una fila de personas esperando en una taquilla, en la que la persona que llega primero es la primera en ser atendida y en la que se agregan nuevos miembros al final de la fila.

En resumen, las colas son una estructura de datos muy útil en programación que se caracterizan por seguir el principio FIFO y que se utilizan para almacenar y gestionar un conjunto de elementos en el orden en que fueron agregados. Sus aplicaciones:

- Validación de paréntesis: Las pilas se utilizan para verificar si una expresión matemática o una expresión regular tiene una sintaxis correcta en cuanto a la apertura y cierre de paréntesis.
- Inversión de cadenas: Las pilas se utilizan para invertir el orden de los caracteres en una cadena.
- Implementación de la navegación hacia atrás en los navegadores web: Las pilas se utilizan para mantener un registro de las páginas web que se han visitado para permitir la navegación hacia atrás.
- Gestión de la memoria: Las pilas se utilizan en los sistemas operativos para gestionar la memoria de un programa.

MATERIAL Y EQUIPO

El proyecto se lleva a cabo con los debidos requisitos necesarios. Contar con una computadora con los componentes adecuados, capaz de soportar IDES de programación con lenguaje de java. Se necesita de software específico para la creación de código en Java, el cual puede ser Eclipse o NetBeans. Se cuenta además con los conocimientos otorgados por la profesora en clase, además de material de apoyo para reforzar el conocimiento.

DESARROLLO Y RESULTADOS

Para la siguiente práctica, debemos saber que se utilizaran datos genéricos, por lo que, tanto en la interfaz como en las clases, se debe de aclarar que se necesitan datos genéricos (<T>). Utilizaremos los materiales antes necesarios, con la versión de Eclipse mas reciente a la fecha de creación de este reporte. Para no extender el reporte, se utilizarán solo los códigos mas relevantes para las pilas y colas, y se describirán con imágenes para ser más claros.

Antes que nada, comenzamos creando interfaces para los métodos en común que tendrán nuestras clases, los cuales son los siguientes:

1. Interfaz de pila

```
package PilaEstatica;

public interface PilaTDA <T>{
    public boolean isEmpty();//Regresa true si la pila no tiene elementos
    public T pop();//Debe quitar el elemento que esta en el tope y regresarlo
    public void push(T dato);//Inserta el dato en el tope de la pila
    public T peek(); //Regresa el elemento que esta en el tope, sin quitarlo
    //public void freePila(); //Limpia pila


}
```

2. Interfaz de Cola

```java
package ColaEstatica;

public interface ColaTDA <T>{
    public boolean isEmpty();
    public void push(T dato);
    public T pop();
    public T peek();
    }
```

3. Interfaz de listas enlazadas (En proceso)

```java
package OperacionesListas;

public interface OperacionesI <T>{
    public void altas(T dato);
    public void bajas(T dato);
    public boolean validaVacio();
    public String imprime();
    public byte buscar(T dato);
    public void modifica(T dato);
}
```

- PILAS

  En primer lugar, comenzaremos con pilas, para eso creamos una nueva clase, puede notarse que se implementa la interfaz con datos genéricos:

```java
package PilaEstatica;
import EntradaSalida.tools;

public class PilaA<T> implements PilaTDA<T> {
private T pila[];
private byte tope;
```

Seguido de eso, crearemos los métodos con el mismo nombre que tienen los métodos de nuestra interfaz:

```
public PilaA(int max)
{
    pila=(T[])(new Object[max]);
    tope=-1;
}

public boolean isEmpty()
{
    return(tope==-1);
}
public boolean isSpace()
{
    return (tope<pila.length-1);
}
public void push(T dato) {
    if(isSpace()) {
        tope++;
        pila[tope] = dato;
    }
    else
        tools.ImprimeErrorMsje("PILA LLENA");
}
public T pop() {
    T dato = pila[tope];
    tope--;
    return dato;
}
```

```
public T peek() {
    return pila[tope];
}
public String toString() {
    return toString(tope);}

private String toString(int i) {
    return(i>=0)?"\n"+"tope["+i+"]==>"+""+pila[i]+""+toString(i-1):"";

}

}
```

Debemos tener en cuenta lo siguiente: El método push almacena un dato, el método pop elimina el dato que se encuentra en el tope de la cima, el método peek nos permite visualizar el dato en el tope de la cima, sin eliminarlo, y el método Free permite vaciar la pila, para ingresar nuevos datos.

- COLAS

  Para los códigos de colas, se crea nuevamente una clase, con datos de tipo genérico:

```
package ColaEstatica;

import EntradaSalida.tools;

public class ColaA <T> implements ColaTDA<T>{
    private T cola[];
    private byte u;

    public ColaA(int max) {
        cola=(T[])(new Object[max]);
        u=-1;
    }
```

  En seguida se crean los métodos correspondientes a la interfaz, en el caso de las colas, algunos métodos vienen definidos por parte de java, y tan solo debemos invocar el nombre del método definido que usaremos, pero en este caso usaremos otro ejemplo para un mejor entendimiento del proceso:

```
public boolean isEmpty() {
    return (u==-1);
}
public boolean isSpace() {
    return(u<cola.length-1);
}
public void push(T dato) {
    if(isSpace()) {
        u++;
        cola[u]=dato;
    }
    else tools.ImprimeErrorMsje("Estructura cola llena");
}
public T pop() {
    T dato= cola[0];
    for(int i=cola.length-1; i>u; i--){
            cola[i]=cola[u+1];
    //invocar a corrimiento inmediato
    //metodo de recorrer del tema 1, cola de u+1
    }
          u--;
    return dato;
}
public T peek() {
    return (T)cola[0];
}
public String toString() {
    return toString(0);
}
private String toString(int i) {
    return (i<=u)?"u===>"+i+"["+cola[i]+"]   "+toString(i+1):"";
}
```

Para los métodos anteriores es el mismo caso que en las pilas: push agrega un dato, pop elimina el dato en el tope de la cima, mientras que peek únicamente lo visualiza, y por último free nos funciona para vaciar los datos de la cola.

- MENU PRINCIPAL

  Para el caso de la clase ejecutable, se presentará el menú de las colas, ya que, los dos tienen similitudes y las diferencias son solo en algunas líneas de código y nombres de los objetos.

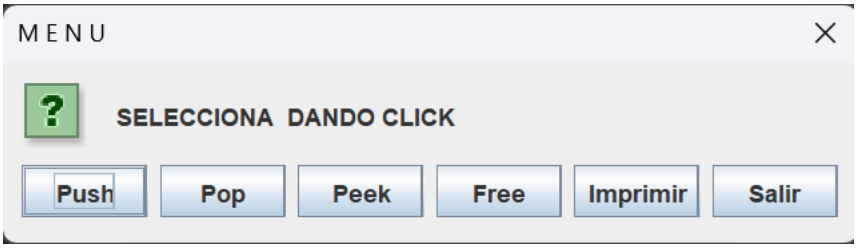
```java
package ColaEstatica;
import EntradaSalida.tools;
public class TestmenuCA {

    public static void main(String args[])
    {
        menu3("Push,Pop,Peek,Free,Imprimir,Salir,");
        }

    public static String boton(String menu)
    {
        String valores[] = menu.split(",");
        int n;
        n=JOptionPane.showOptionDialog(null,"SELECCIONA DANDO CLICK","PILAS",JOptionPane.INFORMAT
                ,valores,valores[0]);
        return (valores[n]);
    }
    public static void menu3(String menu)
    {
        ColaA<Integer>cola= new ColaA<Integer>(10);
        String op;
```

```java
        ColaA<Integer>cola= new ColaA<Integer>(10);
        String op;
        do {
            op=tools.boton(menu);

            switch(op) {
            case "Push":
                cola.push(tools.LeerEntero("Dame un valor"));
                tools.ImprimeMsje(""+cola.toString());

                break;
            case "Pop":
                if(cola.isEmpty()==false) {
                tools.ImprimeMsje("Dato eliminado: "+cola.pop());
                }
                else {
                    tools.ImprimeErrorMsje("PILA VACIA");
                }
                break;
            case "Peek":
                if(cola.isEmpty()==false)
                tools.ImprimeMsje("Dato en el tope:"+cola.peek());
                else tools.ImprimeErrorMsje("COLA VACIA");
                break;
            case "Free":
                if(cola.isEmpty()==false) {
                new ColaA<Integer>(10);
                tools.ImprimeMsje("COLA VACIADA");
                }
                else {tools.ImprimeErrorMsje("COLA VACIA");
                }
                break;
```

Para este caso, se utilizan cases para ejecutar cada método que el usuario requiera, tanto para pilas y colas la estructura es la misma, cambiando nombres de alguno que otro método y nombres de objetos.

- RESULTADOS

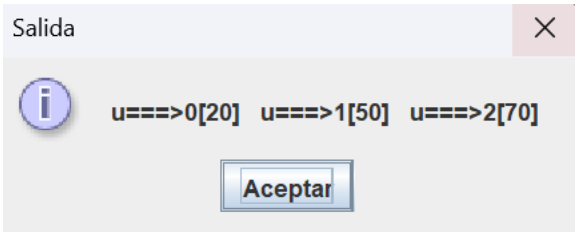
  El resultado será un menú, el cual contiene las opciones Push, Pop, Peek, Free, Imprimir y salir:



  Al agregar datos, para Pilas y Colas, las posiciones de los datos cambiaran:

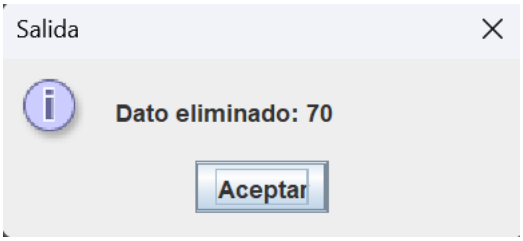
  Pilas

  Colas

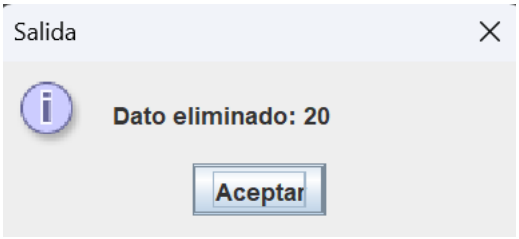


  Para la eliminación de datos mediante push, Pilas eliminara el ultimo dato en ser agregado, mientras que Colas eliminara el primero en ser agregado:

  Pilas



  Colas



CONCLUSION

Al termino de esta práctica, podemos afirmar que las Pilas y Colas nos sirven para almacenar datos, pero cada una trabaja de manera diferente, ya que, en las pilas, el primer dato es el ultimo en salir, mientras que, en las colas, el primer dato es el primero en salir, por lo que hay que tener en cuenta esto a la hora de desarrollar código, ya que, pueden sernos de mucha ayuda si las diferenciamos correctamente y las sabeos aplicar. De igual forma, podemos saber que las interfaces nos ayudan a saber que métodos debe cumplir obligatoriamente un código, ya que, al no haber los especificados en la clase, se producirá un error que no nos dejará continuar con la ejecución del programa.

BIBLIOGRAFIAS

- David S. 21/sep/2018. Estructuras de datos: diferencias entre pilas y colas. https://www.cuasar.net/guia-de-inicio/estructuras-de-datos-diferencias-entre-pilas-y-colas.php
- Rene C. 2015. Pilas y Colas. https://ccc.inaoep.mx/ingreso/programacion/corto2015/Curso-PROPE-PyED-5-Pilas-Colas.pdf
- S/N. S.F. 11.4 Estructuras de datos básicas. https://pier.guillen.com.mx/algorithms/11-otros/11.4-estructuras_datos.htm
- S/n. S.F Pilas y Colas. Udec. http://www.inf.udec.cl/~jlopez/FUNDPRO/apuntesC/clase12.html
- S/N. 18/nov/2015. Pilas y Colas. Programación-Estructura de datos. https://estructura-de-datos-rodrigo.blogspot.com/2015/11/pilas-y-colas.html